СИСТЕМЫ ТЕХНИЧЕСКОГО ЗРЕНИЯ

LU – РАЗЛОЖЕНИЕ. БИБЛИОТЕКА EIGEN.

СЛАУ – ЧТО ЕСТЬ КРОМЕ ГАУССА (И ЗАЧЕМ ЭТО НУЖНО)

- Рассмотрим метод Гаусса с более общих позиций

$$A \cdot x = b$$

- $A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$

- Попробуем рассмотреть операцию исключения элементов первого столбца:

$$A^{(1)} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & & a_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

- Для этого мы вычисляли коэффициенты $\mu_{21}, \mu_{31}, \dots, \mu_{n1}$

ИСКЛЮЧЕНИЕ СТОЛБЦА – УМНОЖЕНИЕ НА МАТРИЦУ

- Давайте введём матрицу

$$M_1 = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ -\mu_{21} & 1 & 0 & \dots & 0 \\ -\mu_{31} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\mu_{n1} & 0 & 0 & \dots & 1 \end{pmatrix}$$

- Можно заметить, что $A^{(1)} = M_1 \cdot A$
- Аналогично

$$M_2 = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & -\mu_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -\mu_{n2} & 0 & \dots & 1 \end{pmatrix}$$

ПРИВЕДЕНИЕ К ВЕРХНЕТРЕУГОЛЬНОМУ ВИДУ

- В результате получим $A^{(n-1)} \cdot x = b^{(n-1)}$, где

$$A^{(n-1)} = M_{n-1}M_{n-2} \dots M_2M_1A$$
$$b^{(n-1)} = M_{n-1}M_{n-2} \dots M_2M_1b$$

- Можно выразить и в обратном направлении:

$$A = M_1^I M_2^I \dots M_{n-2}^I M_{n-1}^I A^{(n-1)}$$

при этом

$$M_1^I = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ \mu_{21} & 1 & 0 & \dots & 0 \\ \mu_{31} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_{n1} & 0 & 0 & \dots & 1 \end{pmatrix}$$

и т.д.

LU-РАЗЛОЖЕНИЕ

- Перемножим матрицы:

$$M_1^I M_2^I \dots M_{n-2}^I M_{n-1}^I = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ \mu_{21} & 1 & 0 & \dots & 0 \\ \mu_{31} & \mu_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_{n1} & \mu_{n2} & \mu_{n3} & \dots & 1 \end{pmatrix} = L, \qquad A^{(n-1)} = U$$

- И, в результате, $A = LU$

$$Ax = LUx = b$$

- Здесь L – *нижнетреугольная* матрица, а U – *верхнетреугольная* матрица

LU-разложение

- В результате метод Гаусса можно разделить на 3 части:

Вычисляем матрицы L и U

Вычисляем $b^{(n-1)}=L^{-1}b$

Решаем Ux=$b^{(n-1)}$

LU-РАЗЛОЖЕНИЕ

- На первый взгляд мы усложнили себе жизнь
- Но, на самом деле мы просто выделили преобразование $b^{(n-1)}$
- За счёт этого мы можем повторять этапы 2 и 3 для РАЗНЫХ правых частей
- На практике метод Гаусса всегда реализуется через ***LU*-**разложение
    - Не зависит от правой части, а только от матрицы
    - Для многих ситуаций правая часть и не нужна (определитель, обратная матрица и т.д.)
    - Если же нам нужно решение – просто используем уже найденное разложение

РАЗЛИЧНЫЕ МАТЕМАТИЧЕСКИЕ БИБЛИОТЕКИ ДЛЯ C/C++

- C
  - Intel Math Kernel Library (MKL) – очень оптимизированная проприетарная
  - GNU Scientific Library (GSL)
- C++
  - Armadillo
  - Blitz+
  - Boost.uBLAS
  - Eigen
  - Intel MKL
  - MTL4
  - …

EIGEN

- Пакет для матричных вычислений
- Матрицы, их преобразования, нормы и т.п.
- Методы решения СЛАУ
- Кватернионы, углы Эйлера и т.д.
- Лицензия LGPL
- Header-only библиотека (ставится на любую ОС)
- Доступен в большинстве дистрибутивов Linux и MacOS (через Homebrew)

eigen.tuxfamily.org

КАК ПОДКЛЮЧИТЬ EIGEN

- Eigen – так называемая header-only библиотека
- Всё, что нам нужно сделать – это указать путь до её заголовочных файлов
- В различных UNIX'ах Eigen ставится через пакетные менеджеры
  - В этом случае – путь по-умолчанию
- Под Windows (или установке через Интернет) – путь будет отличаться

ПРОЕКТНЫЙ ФАЙЛ

```
TEMPLATE = app
CONFIG += console c++11
CONFIG -= app_bundle
CONFIG -= qt
# Вот указание пути до библиотеки
INCLUDEPATH += /usr/include/eigen3
#INCLUDEPATH += \
#   C:\projects\eigen-eigen-323c052e1731

SOURCES += \
        main.cpp
```

<EIGEN/CORE> МАТРИЦЫ…

- Все элементы размещены в пространстве имён Eigen
  - Либо указывать явно
  - Либо подключать пространство имён
- Основной объект – матрицы

Eigen::Matrix{N}{t}

- N – размер матрицы:
  - 2 – для матриц 2x2
  - 3 для матриц 3x3
  - 4 для матриц 4x4
  - X для матриц *динамически* определяемого размера
- t – тип элементов: {i}nteger, {f}loat, {d}ouble
- Например Matrix4i, Matrix2d, MatrixXf

<EIGEN/CORE> …И ВЕКТОРА

- Vector{N}{t} – вектор-столбец
  - Параметры – аналогично матрице
  - Примеры Vector3d, Vector2f, VectorXd
- RowVector{N}{t} – вектор-строка
  - То же самое
  - RowVectorXd и т.

Eigen различает строки и столбцы!!!

EIGEN

```
#include <iostream>
#include <Eigen/Core>

using namespace std;
using namespace Eigen;

int main()
{
    Eigen::Matrix3d mat;
    mat << 3,  2,  4,
           0,  0.1, 2.3,
           0, 5.2, 4.3;
    Vector3d vec;
    vec(0) = vec(1) = 0;
    vec(2) = 1;


    cout << "Matrix" << endl << mat << endl;
    cout << "Vector" << endl << vec << endl;
    cout << "Multiplication result "  << endl << mat*vec << endl;
    //  cout << "Multiplication result "  << endl << vec*mat << endl;
    // Ошибка: неправильный размер
    RowVector3d vec2 = vec;
    cout << "Row miltiplication result "  << endl << vec2*mat << endl;
     cout<<"Row x collumn multiplication " << vec2*vec <<endl;
     cout<<"Collumn x row multiplication « << endl<< vec*vec2 <<endl;
```

•

РЕЗУЛЬТАТ

- Умножение столбца на строку – матрица
- Умножение строки на столбец - скаляр

```
Matrix
  3   2   4
  0 0.1 2.3
  0 5.2 4.3
Vector
0
0
1
Multiplication result
  4
2.3
4.3
Row miltiplication result
  0 5.2 4.3
Row x collumn multiplication 1
Collumn x row multiplication
0 0 0
0 0 0
0 0 1
```

ИНДЕКСНОЕ ОБРАЩЕНИЕ

КОД

```
MatrixXd mat(4,4);
for (int i = 0; i<mat.rows(); i++)
  for(int j = 0; j<mat.cols(); j++)
     mat(i,j) = i+j;
cout << mat << endl;
```

РЕЗУЛЬТАТ

```
0 1 2 3
1 2 3 4
2 3 4 5
3 4 5 6
```

ВЫДЕЛЕНИЕ БЛОКОВ

КОД

```
MatrixXd mat(4,4);
for (int i = 0; i<mat.rows(); i++)
   for(int j = 0; j<mat.cols(); j++)
        mat(i,j) = i+j;
MatrixXd mat2 = mat.block(1, 1, 2, 2);
MatrixXd mat3 = mat.block<2,2>(1, 1);
cout << mat3 << endl;
```

РЕЗУЛЬТАТ

```
2 3
3 4
```

<EIGEN/DENSE>
ЗАПОЛНЕННЫЕ МАТРИЦЫ

- Мы можем делать с заполненными векторами и матрицами целую кучу операций:
- Арифметические
- Скалярное / векторное произведение
- Вычисление средних, дисперсий…

ВЕКТОРНЫЕ И СКАЛАРНЫЕ ПРОИЗВЕДЕНИЯ

КОД

```
#include <Eigen/Dense>
using namespace std;
using namespace Eigen;

Vector3d a, b;
a << 1, 0, 0;
b << 0, 1, 0;

cout << "Dot product "<< a.dot(b) << endl;
cout << "Cross product "<< endl << a.cross(b) << endl;
```

РЕЗУЛЬТАТ

Dot product 0

Cross product

0

0

1

КОД

```
#include <Eigen/Dense>
Matrix2d mat;
mat << 1, 2, 3, 4;
cout << "Here is mat.sum(): " << mat.sum() << endl;
cout << "Here is mat.prod(): " << mat.prod() << endl;
cout << "Here is mat.mean(): " << mat.mean() << endl;
cout << "Here is mat.minCoeff(): " << mat.minCoeff() << endl;
cout << "Here is mat.maxCoeff(): " << mat.maxCoeff() << endl;
cout << "Here is mat.trace(): " << mat.trace() << endl;
```

РЕЗУЛЬТАТ

```
Here is mat.sum():       10
Here is mat.prod():      24
Here is mat.mean():      2.5
Here is mat.minCoeff():  1
Here is mat.maxCoeff():  4
Here is mat.trace():     5
```

EIGEN - LU

```
#include <iostream>
#define _USE_PREDEFINED_CONSTANT
#include <cmath>

#include <eigen3/Eigen/LU>
#include <eigen3/Eigen/Dense>
using std::cout;
using std::endl;

int main()
{
    Eigen::MatrixXd a(4,4);     //  Матрица бибиотеки Eigen

    a << 10, 6, 2, 0, 5, 1, -2, 4, 3 ,5, 1, -1, 0, 6, -2, 2;
    cout << "Matrix A\n" << a << endl;
    Eigen::VectorXd b(4), x(4); //  Вектор - матрица с одним столбцом
    b << 25, 14, 10, 8;
    cout << "Vector b:\n" <<b << endl;

    x = a.lu().solve(b);
    cout << "Solution \n"<< x << endl;

    Eigen::VectorXd res = a*x-b;
    cout << "Residual vector\n" << std::fixed <<res << endl;
    cout << "Norm of residual vector " <<res.norm() << endl;


    return 0;
}
```

EIGEN - LU

```
Matrix A
10  6  2  0
 5  1 -2  4
 3  5  1 -1
 0  6 -2  2
Vector b:
25
14
10
 8
Solution
   2
   1
-0.5
 0.5
Residual vector
0.000000
0.000000
0.000000
0.000000
Norm of residual vector 0.000000
Press <RETURN> to close this window...
```